J Hardw Syst Secur (2017) 1:38–55
DOI 10.1007/s41635-017-0002-5

Security Assurance Guidance for Third-Party IP

Brent Sherman[1] · **Mike Borza**[2] · **Brian Rosenberg**[3] · **Charles Qi**[4]

Received: 14 November 2016 / Accepted: 20 March 2017 / Published online: 7 April 2017

Abstract System OEMs are increasingly adopting the motto "Trust but verify" when it comes to their supply chains. After several public incidents in which trusted vendors unknowingly provided vulnerable components, OEMs are requesting evidence of security assurance before integrating components into their products. It can be problematic for semiconductor vendors to provide such evidence since their products often contain 3rd party components that are typically treated as black boxes. Moreover, asking 3rd party vendors to provide such evidence for their components is equally problematic due to the many integration unknowns and a lack of applicable literature on security assurance for standalone technologies. We address these issues by defining a security process and relationship between semiconductor vendors and trusted 3rd party component providers and a practical methodology to produce standardized quality security assurance evidence. We provide example applications of the methodology using several open source components.

Keywords Threat model · Security assurance · Security development lifecycle · Commercial Off-the-Shelf (COTS) · IP development

✉ Brent Sherman
brent.m.sherman@intel.com

[1] Intel Corporation, Hillsboro, OR, USA

[2] Synopsys, Inc, Kanata, Ontario, Canada

[3] Qualcomm Technologies, Inc., San Diego, CA, USA

[4] Cadence Design System, Inc., Santa Clara, CA, USA

1 Introduction

There have been several incidents in which system OEMs have suffered costly security vulnerabilities due to 3rd party components [1–4]. As far as we are aware, all were caused by software components, making the mitigation of these vulnerabilities relatively simple (e.g., patch, disablement, or removal). However, mitigating a similar problem arising from insecure hardware would require an expensive recall. To help manage security risk, OEMs are requesting hardware vendors to provide evidence of security assurance to increase confidence in a component's quality. In general, semiconductor vendors will be able to accommodate these requests, since they know the data flows, use cases, interdependencies, etc. of their silicon, all of which are needed to perform security assessments using existing methodologies [5–8]. However, if the vendor has integrated 3rd Party Intellectual Property (3PIP, IP), their knowledge alone would typically not be sufficient to produce quality security assurance evidence. Most semiconductor vendors view 3PIP as black box technology that hooks into their silicon and "just works." Furthermore, 3PIP is often provided only as a netlist, which limits what security evaluation a semiconductor vendor (now functioning as an Integrator) can perform. An Integrator can perform security design checking [18] and/or formal verification; however, these can be time consuming and still not address all integration security concerns. Therefore, Integrators depend on trusted IP providers to produce security assurance collateral for their technologies. However, existing security assurance methodologies require system level information in order to complete. Unfortunately, this information is rarely available to IP providers, mainly because 3PIP is designed, developed, and productized well before Integrators define their product requirements. It is not uncommon for Commercial

Off-the-Shelf (COTS) IP to be developed several years before product integration. We address this dilemma by defining an industry security assurance process at the IP level, the types of collateral information to be produced, and how to produce this collateral information. We also provide examples of how this process can be applied to some selected open source IPs.

In summary, the contributions of this paper are as follows:

- Definition of a process between an end-product Integrator and a trusted IP vendor to capture and use standardized security assurance collateral to communicate potential security concerns related to integration of IP. Over time, the process can be used to assess the competence of an IP vendor for establishing or increasing trust.
- Creation of a practical methodology, focused on COTS IP integration, to produce quality security assurance evidence which includes (1) a security risk assessment that identifies today's known integration security concerns, along with actions and recommendations to reduce risk and (2) a light-weight IP integration threat model.
- Formulation of a common language between COTS IP providers and Integrators to communicate potential security concerns in IP integration. This common language makes it efficient and cost-effective for IP providers to support multiple customers and recognizes the need of integrators who work with multiple IP providers to be able to provide security assurance for their products to system OEMs.

The paper is structured as follows: Section 2 outlines the overall process between an IP provider and Integrator. Section 3 describes the methodology and types of collateral needed for usable security assurance evidence. Section 4 shows how the threat modeling methodology can be applied using open source cores as examples. We lay out our Conclusion in Section 5 and provide an Appendix A with figures detailing the methodology outlined in Section 3.

2 Overall Process Flow

The IP security assurance process flow between an end-product Integrator and a trusted IP provider is shown in Fig. 1. It is based on both parties sharing the goal of producing high-quality secure products, which are common in the semiconductor industry (i.e., integrators do not engage non-reputable or untrusted vendors). Additionally, the collateral produced should not be considered a replacement for formal verification or similar practices[18]. The process focuses on COTS products and not co-designed or co-developed IP. The process also aligns with architecture and design requirements in a typical Security Development Life Cycle (SDL) process [6]. Lastly, Fig. 1 depicts a minimal flow that may be expanded to accommodate custom engagements and/or other stages of the SDL process.

The dashed rectangle on the left labeled "Assessment" encloses collateral that is expected for all COTS IP a vendor produces. Note that the IP for which collateral is provided does not necessarily have explicit security objectives or claims; thus, the process is designed to accommodate IP providers that are experts in their functional areas, but do not necessarily have security expertise. Due to the potentially sensitive nature of the collateral, IP providers may choose to protect it with a confidentiality agreement, but it may also be provided freely under the general IP license. The Integrator reviews the collateral to understand any security concerns identified (and those areas of concern that might

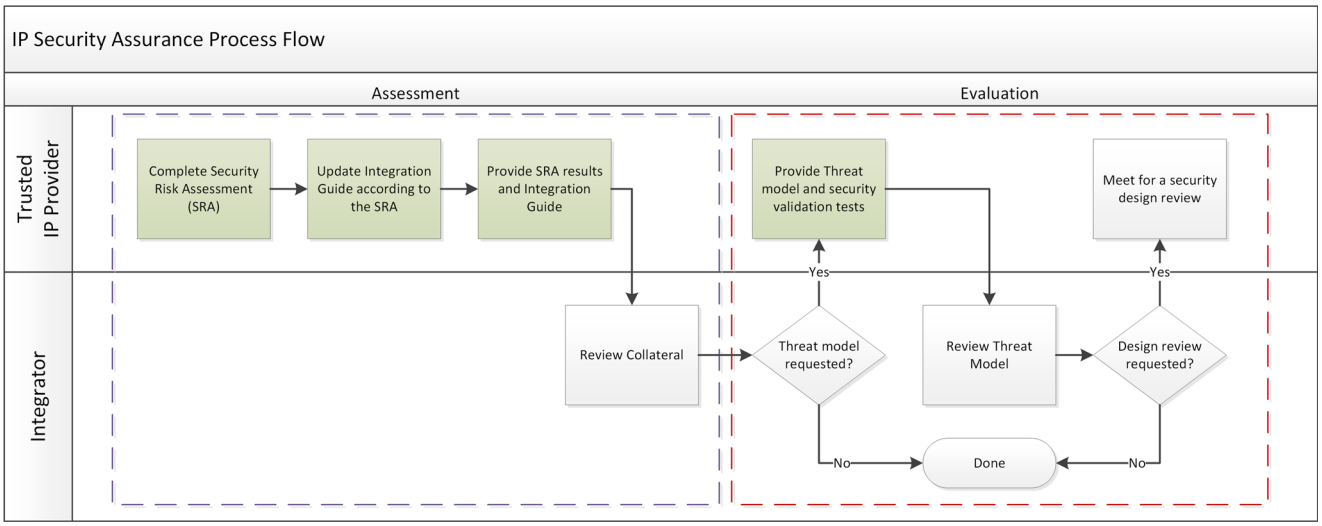


Fig. 1 IP security assurance process flow

not have been identified) and the conditions under which the IP can be properly integrated into the product. Based on this review, the Integrator may decide to (1) not to use the IP and possibly engage with another vendor or (2) accept the IP and integrate or (3) request a threat model for further evaluation. The conditions for each outcome are Integrator and application specific and out of scope of this process.

The request for threat modeling collateral is enclosed by the dashed rectangle on the right labeled "Evaluation." The threat model should highlight security concerns that an Integrator needs to be aware of, and Integrators must decide which concerns are relevant to their product. In most cases, the process will stop here, meaning security concerns have been addressed. When there are particular concerns with some threats, the Integrator may wish to conduct a security design review with the IP vendor. This is a joint review in which the IP vendor and Integrator examine the IP design in greater detail, in particular the implementation of mitigations for one or more threats. Since this kind of review is usually quite specific to the IP and operational domain, this process is also out of scope for this paper. The important point is that the process that identified a concern stems from the methodology that is the subject here. The remainder of this paper focuses on the details of the green boxes in Fig. 1.

3 Methodology

This section details what security assurance an IP provider should produce in order to address the Integrator's security concerns. There are two main components (though there may be others as well): (1) Security Risk Assessment and ensuing Integration Guidance and (2) Architectural and Design Threat Model.

3.1 Security Risk Assessment

The goal of this activity is to perform a Security Risk Assessment (SRA) on the architecture and design of the IP. The SRA consists of a list of questions that will help identify today's known security concerns [8]. There are currently 18 topics such as debug access, test modes, and access protections; however, the list may grow as new areas of concern are identified. The SRA expands on [8] by providing practical guidance that lists concerns, actions, and/or recommendations on which an Integrator may act. Each question is designed to have either a follow-on question to drill down further or provide actionable guidance to address the concern. The associated concern is the reason why the question is being asked. For example, a question about debug registers is concerning because often these registers can be used maliciously (e.g., to bypass existing protections or allow unrestricted assess to an asset). Often, an action will be identified for the IP provider to complete. All actions should be completed before an Integrator considers using the IP in a product. In the debug register example, one action would be to label these registers with a `DEBUG` tag in the Integration Guide so to get the proper attention. Lastly, a recommendation may be provided for the Integrator to consider. In the same example, a recommendation would be to restrict access to these debug registers by implementing access control protections at the integration layer. Table 1 defines a few terms that are useful in understanding the SRA. Appendix A provides full details of each question and its path flows, starting with Fig. 8. To close, it is worth emphasizing that SRA results should not be used to represent overall risk of an IP since this can often lead to a false sense of security, as proven in [8].

Table 1 SRA terms and assumptions

Item	Description
Native vs external	The SRA is intended to address security concerns at both the IP and integration level. To differentiate between these two levels, the terms native and external are used. *Native* refers to concerns that arise within the IP itself, while *external* refers to anything outside it
Security review	For the purpose of this document, a security review refers to, at a minimum, a threat model which is defined in Section 3.2. However, additional collateral such as validation and/or verification tests, code review results can be included for completeness

3.2 IP Threat Model

After reviewing the results of the SRA, an Integrator may request the IP vendor to provide a threat model. This will typically occur when there is assertion of a concern listed in the SRA. The goal of the threat model is to elucidate security threats and concerns to which the IP may be exposed in order to provide guidance to the integrator. The threat model attributes are shown in Table 2 and are focused around a functional asset-based approach [9]. The benefits of this

Table 2 IP threat model attributes

Attribute	Definition
Asset	Anything of value or importance that is crucial to proper behavior. Assets are typically contained within the IP
Access point	Any means by which an adversary can gain access to an asset
Threat	Anything that can potentially alter or compromise the behavior of an asset
Concern	The potential harm that a threat poses to an asset
Mitigation	An action or countermeasure put in place to reduce the severity of a threat

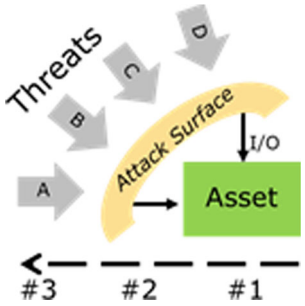


Fig. 2 Asset-centric approach

approach are (1) it is straightforward and easy to implement by non-security individuals and (2) functional behaviors often persist in integration regardless of system complexity whereas vulnerabilities found by other approaches may not. The attributes in Table 2 should be completed for each identified {threat, asset} pair.

As shown in Fig. 2, we recommend an inside-out approach as follows:

1. Identify the assets
2. Identify the access points to the assets
3. Identify the threats that are exposed by the access points and list the concerns and mitigations associated to each threat

The process is complete once all the access points for each asset have been examined for threats and the results have been documented. The following sections provide more details on each step.

3.2.1 Asset Identification

Probably, one of the more difficult steps in IP threat modeling is identifying the assets. This step by nature is subject to ambiguity. The reason is because many consider either "everything" or "nothing" in an IP as an asset. This mindset of all or nothing can often lead to an incorrect or unusable threat model (i.e., too many false positives). To help minimize the confusion, focus on functional objectives and claims of the IP instead of a particular security concern(s). Addressing security concerns will be part of step 3 in the methodology. To find the functional objectives, answer the question "What exactly is the functional purpose of the IP?" Next, answer "What minimal components are required to achieve it's functional purpose?" This exercise will help resolve what the true assets in the IP are.

3.2.2 Access Points Identification

Once an asset has been identified, finding its access points is fairly straightforward. Access points can be either directly or indirectly connected. The question to address is what interfaces (e.g., signal, pin, bus, register) can be used to gain access to (i.e., alter or compromise the behavior of) an asset. Often access points can be referred to as vulnerabilities.

A subtle point about access points is that the complete set may not be immediately obvious to the IP provider. Many IP providers supply their product as RTL source code or netlist. As part of the integration process, new access points are created via scan insertion and other processes that are designed to enhance manufacturing, testability, etc. Some assets may become visible by leakage through side channels such as power or electromagnetic emanation signatures. While these access points are primarily of concern to security IP providers (who should be aware of them), they might provide access to any asset. Complete identification by the IP provider of assets will enable the integrator to determine any threats associated with these.

3.2.3 Threats, Concerns, and Mitigations

The last step in the process is to identify any threats and associated concerns and, if possible, provide a potential mitigation. Once a threat is identified, at least one concern must be associated with it. This concern will highlight what is at risk to the asset due to the threat. This information is critical to the Integrator since it will determine which threats will be in scope for the product. Not all threats may be in scope once the IP is integrated, as we explain in more detail later in this section. Any threats that are mitigated should be noted as such. If the IP itself does not include a mitigation, a recommendation can be provided for how to mitigate the threat in integration.

It is not a requirement that the IP provider implement mitigations to all identified threats. This is because, as noted above, not all threats may be in scope for a product, or the mitigation may be better implemented at the integration level. Unless the threat is associated to a specific objective or claim that the IP promotes, mitigations should be an activity for the Integrator to implement. The main motivation for this approach is help the IP provider keep logic, footprint, power, and cost to a minimum.

There are several situations in which including a mitigation within an IP can lead to unnecessary effort. Such an example might be an IP that supports multiple bus interfaces, each with its own set of potential threats that are mitigated by additional logic. However, the Integrator only plans to use one of those buses, leaving the rest unconnected. In this case, the IP provider expended unnecessary effort and added unnecessary logic to mitigate threats that are not in scope for the product. In a different vein, sometimes a single mitigation implemented at the integration level can thwart a threat across multiple IPs. Suppose that a particular malformed packet impacts several IPs on a peripheral bus. Instead of each IP blocking this packet, a better mitigation would be to prevent it from making it onto

the bus altogether, which is an integration activity. Lastly, product constraints may render IP-level mitigations ineffective. Suppose bit 31 of a configuration register serves as a lock to prohibit write access to a register range. However, if the integrating product only supports a 16-bit single-access data bus, then this lock mitigation would be unreachable. Of course one could argue that the real problem is that this IP should not be integrated into this type of product. Nevertheless, the reality is that COTS IP providers have no control over how an IP is integrated into a product.

To help guide whether a mitigation should be implemented within the IP, consider these following questions: (1) Can the mitigation be implemented ONLY in the IP? (2) Is the threat in scope for typical implementations AND is the mitigation BEST implemented in the IP? If either answer is yes, the IP should include the mitigation; otherwise, it should be noted and left for the Integrator.

3.2.4 Threat Analysis Omission

For products in which system and data flows are understood from end to end, a threat analysis is typically expected for each threat identified. Methods such as STRIDE and DREAD [10] are used to understand the risk and severity of threats. This activity may be almost impossible to perform for COTS IP since such product flows are unknown during the time of development. Therefore, it is recommended that the threat analysis be performed by the Integrator, not the IP provider.

3.3 Complete Walk-Through Examples

This section illustrates the complete process described in Fig. 1 by walking through the design of two open core examples: (1) RC4 crypto core and (2) Watchdog timer. The cores were selected to highlight the complete methodology because the implementations are simple, publicly available in source RTL and well documented, the security claims made are transparent so as not to overshadow the process, and the collateral produced is minimal, making it easier to comprehend.

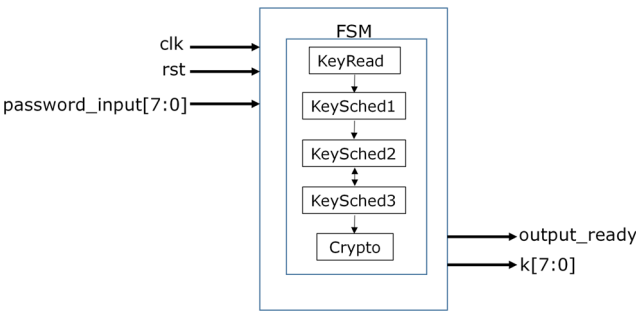


Fig. 3 RC4 core

Table 3 RC4 SRA results

Question	Answer	Comments
Q1–Q3	No	–
Q4: Save/restore	No	If power is lost in the middle of a key generation, the key is discarded. Once the key is created, the Integrator is responsible for saving and restoring it as needed
Q5: Random numbers	No	For proper RC4 operation, the *password_input* should be unique per stream
Q6: Integrity checking	No	–
Q7: Cryptography	Yes	The core generates a keystream for encryption/decryption
Q7.1	Yes	Threat model provided in the next section
Q7.2	No	–
Q8–Q16	No	–
Q17: Security claims	No	Already addressed in Question #7
Q18: Unknowns	No	All addressed in the threat model

3.3.1 Example I: RC4 Core Architecture

Figure 3 is the block diagram of the RC4 core, which is a pseudo-random stream generator [11]. The IP generates an RC4 stream which can be XOR'ed with data to provide confidentiality. Although the RC4 algorithm has been deprecated and deemed insecure, the implementation provides a good example for applying the process to identify security concerns in IP integration.

The core contains a five-stage finite state machine (FSM) that maps to the RC4 algorithm [12]. The basic operational flow is as follows:

1. When *rst* is low, the core will start sampling the password on *password_input*, one byte per *clk* cycle, until KEY_SIZE is reached. (In our example, KEY_SIZE is hardcoded to 0x7.)
2. The FSM performs the key expansion and discards the weak bytes in the stream as defined in RFC 4345.[1]
3. Once the FSM reaches the `Crypto` state and all the weak bytes have been discarded, *output_ready* is asserted to signify that the data on *k* is now valid. The user reads the keystream (*k*), byte-by-byte as needed to encrypt (or decrypt) a message.

3.3.2 Security Risk Assessment

Table 3 shows the answers to the SRA questions for the RC4 core. The only triggers (i.e., "Yes" answers) pertain to the

[1] https://www.ietf.org/rfc/rfc4345.txt

security concerns involving cryptography. Therefore, due to these triggers, the expectation is for the Integrator to request a threat model from the IP provider.

3.3.3 Threat Model

The first step is to identify the IP's functional assets. The purpose of this core is to generate an RC4 keystream. Once taken out of reset, the FSM will generate keystream bytes as long as an external clock is applied and *output_ready* is asserted. Since *output_ready* notifies when a valid key is available, which is controlled by the `Crypto` stage, this stage is the asset. It must be protected from inputs that could cause unwanted behavior (i.e., an invalid key). The next step is to identify the access points to the `Crypto` state. Looking at the block diagram in Fig. 3, the input signals on the left-side of the diagram make up the attack surface. Table 4 is a list of threats that could cause concerns to an Integrator.

Table 4 Threat model for RC4 core

#	Attribute	Definition
1	Asset	`Crypto` stage
	Access point	*password_input*
	Threat	Weak or repeated input will cause a predictable output stream
	Concern	A predictable output stream does not provide adequate confidentiality protection
	Mitigation	None. Recommendations: • Integrator to ensure data on *password_input* is never repeated • Increase the KEY_SIZE defined in the HDL to provide more entropy
2	Asset	`Crypto` stage
	Access point	*clk*, *rst*
	Threat	Manipulate the access points to inject a fault to bypass a state in the FSM [19]
	Concern	The FSM will be in an unwanted or unknown state
	Mitigation	None. This is a sophisticated attack that might be out of scope for an end product. Recommendation: The FSM states are defined by a 4-bit register with only 5 being assigned. In RTL, assigned the 11 undefined states to fail safely/securely
3	Asset	`Crypto` stage
	Access point	None
	Threat	As defined by RFC 4345, the first 1536 bytes of the output keystream are considered weak
	Concern	If the first 1536 bytes are used, the output stream will not provide adequate confidentiality protection
	Mitigation	The FSM discards the first 1536 bytes before *output_ready* is asserted

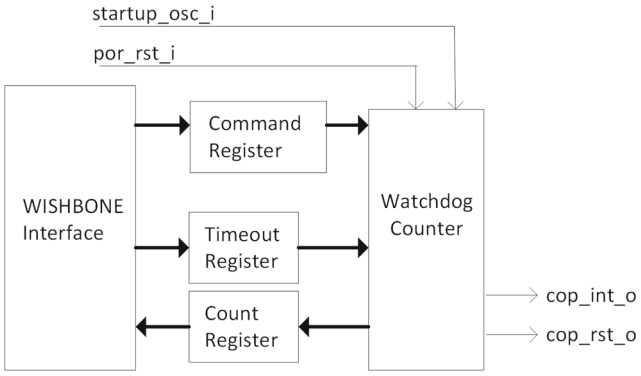


Fig. 4 Watchdog timer[2]

3.3.4 RC4 Conclusion

In summary, the RC4 IP core implementation has relatively low risk due to its limited attack surface and simple design. Other than the insecurity of the RC4 algorithm itself, the IP design has minimal threats. However, there are some things to be considered when integrating to help reduce risk. This becomes apparent in the results of the SRA and threat model. The collateral, which clearly calls out security concerns to address during integration, can be used by the Integrator to improve the quality of their product.

3.3.5 Example II: Watchdog Timer

Figure 4 shows a block diagram for a watchdog timer named Computer Operating Properly (COP) [13]. There is a Watchdog counter component which is configured by three registers that are accessed through a standardized WISHBONE interface. When this counter expires, *cop_rst_o* asserts. The normal operation flow is as follows:

1. Disable the COP by clearing enable bit in the control register (CNTRL.EN).
2. Set the timeout value in the timeout register (TOUT).
3. Enable COP by setting CNTRL.EN. The counter starts decrementing.
4. (optional) Set the configuration protection locks in CNTRL. Once locked, CNTRL settings can not be altered except by reset.

5) Restart the counter by writing 0x5555 and 0xAAAA, in order, to the count register (CNT)

The COP also supports a few non-standard input signals, which are not shown in the diagram. These signals are used for debugging and are listed in Table 5.

[2] http://opencores.org/websvn,filedetails?repname=cop&path=%2Fcop%2Fdoc%2FCOP_specs.pdf

Table 5 COP non-standard signals

Signal	Description
stop_mode_i	System is in STOP mode
wait_mode_i	System is in WAIT mode
debug_mode_i	System is in DEBUG mode
por_rst_i	Power-on reset
arst_i	Asynchronous reset

3.3.6 Security Risk Assessment

Table 6 shows the answers to the SRA questions for the COP core. The tool identified concerns around the non-standard debug signals. Therefore to help the Integrator better understand these concerns, a threat model should be requested from the IP provider.

3.3.7 Threat Model

The functional purpose of this IP is to force a system to a known good state when the watchdog timer expires (i.e., *cop_rst_o* asserts). The core of this functionality is the Watchdog Counter component, and this counter is the asset. In addition to the counter, there is an external clock dependency (*startup_osc_i*), which is critical for operation. However, since this is external to the IP, it will be considered as a dependency instead of an asset. The access points to the counter are the WISHBONE interface and the non-standard input signals. Shown in Table 7 is a list of threats that may be of concern to an Integrator.

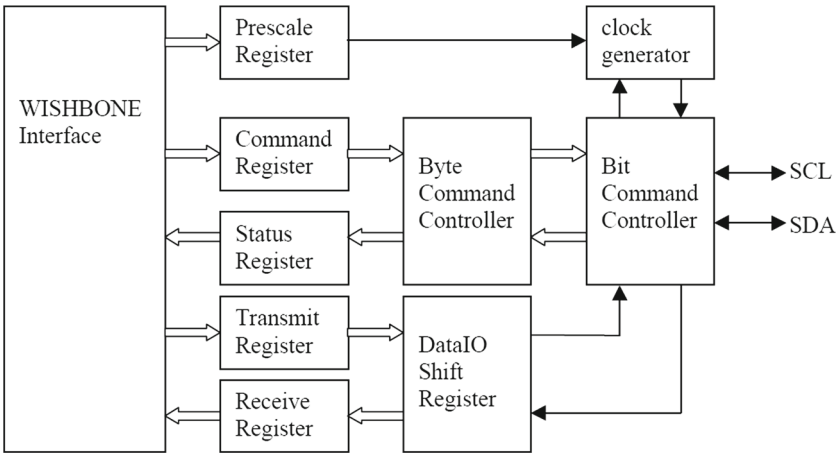
Table 6 COP SRA results

Question	Answer	Comments
Q1: Debug	Yes	No registers, however, there are debug signals that can interrupt the flow of operation
Q1.1	No	No access protection mechanisms on these debug signals
Q2–Q3	No	–
Q4: Save/restore	No	Typical use-case is to have the COP lose state on a power state change. If COP state is required to be restored, the Integrator must provide this functionality
Q5–Q7	No	–
Q8: Access protection	Yes	There are two protection bits to lock the configuration
Q8.1	Yes	Bits 0 & 1 in register CNTRL
Q8.1.1	No	No dependencies
Q9-Q12	No	–
Q13: Non-standard	Yes	There are several non-standard signals that can interrupt normal operational flow, Table 5
Q14–Q18	No	–

Table 7 Threat model for COP

#	Attribute	Definition
1	Asset	Watchdog Counter
	Access point	*stop_mode_i*, *wait_mode_i*, *debug_mode_i*, *por_rst_i*, *arst_i*
	Threat	If the mode is enabled and the corresponding signal is asserted, the Watchdog Counter will stop decrementing
	Concern	The watchdog timer will never expire, disabling the COP's ability to reset a system
	Mitigation	None. Recommendation: At the integration layer, access protection mechanisms should be provided for these non-standard signals to ensure only trusted agents have access
2	Asset	Watchdog Counter
	Access point	CNTRL register (WISHBONE interface)
	Threat	Before the protection locks are set, an attacker can change the configuration settings (i.e. mode settings and timeout value) to prolong the Watchdog Counter expiration or prevent it from decrementing
	Concern	The timer will never expire thus disabling the watchdog's ability to reset a system
	Mitigation	All bit settings in the CNTRL register must be set at the same time by issuing a single write
3	Asset	Watchdog Counter
	Access point	TOUT register (WISHBONE interface)
	Threat	Before the CNTRL.EN bit is set, an attacker can change the timeout value in TOUT to prolong the Watchdog Counter expiration
	Concern	The timer value can be set to the maximum value which delays when the watchdog can reset the system
	Mitigation	None. Recommendation: At the integration layer, access protection mechanisms should be provided for the TOUT register to ensure only trusted agents have access
4	Asset	Watchdog Counter
	Access point	CNT register (WISHBONE interface)
	Threat	The values used to reset the Watchdog Counter can be easily guessed
	Concern	Once the reset values are known, an attacker can reset the counter before it expires, thus preventing the watchdog from resetting the system
	Mitigation	None. Recommendation: Provide access control protections on the CNT register so only a trusted agent can reset the counter

Fig. 5 I^2C master controller[3]



3.3.8 Watchdog Conclusion

The threats identified are not caused by unexpected behavior, meaning the COP functions as architected and designed. However, these behaviors may be undesirable once integrated into a product, especially those of the non-standard signals. By documenting, it informs the Integrator about potential risks that could affect the security robustness of their product.

3.4 Threat Modeling Summary

The threat modeling methodology described in this paper diverges from traditional approaches such as [17] because at the level of a standalone IP, full product data flows, use cases, etc. are unknowns. To overcome this, the methodology eliminates threat analysis and instead focuses on functional threats. As shown in the examples above, often expected IP behavior can yield unwanted results in an end product. Without this methodology, it is unclear how such risks would be identified for an Integrator to take into consideration. To help highlight the benefits of the methodology, additional examples are provided in Section 4.

4 Threat Modeling Examples

This section walks through the architecture and design threat model process for some additional open source IP designs to further highlight how the methodology can be applied. These examples are intended to explain how to apply the process, since it differs from existing literature, and should not be considered absolute or comprehensive. The cores are listed in order of complexity to build a learning progression.

[3] http://opencores.org/websvn,filedetails?repname=i2c&path=%2Fi2c%2Ftrunk%2Fdoc%2Fi2c_specs.pdf

4.1 I^2C Master Controller

Figure 5 shows the block diagram of the I^2C master controller IP [14]. A WISHBONE interface exposes configuration and data registers. Since I^2C is a serial bus, there are two shift controllers, one for data and one for commands. Lastly, the block contains a clock generator and an I^2C signal controller.

4.1.1 I^2C Master Controller - Threat Model

The functional purpose of this IP is to provide an I^2C master interface on the WISHBONE bus. It is basically a pass-through which serializes byte data. This is the job of the

Table 8 Threat model for I^2C master controller

#	Attribute	Definition
1	Asset	Bit Command Controller, Clock Generator
	Access point	PRER - Clock prescale register (WISHBONE)
	Threat	Setting this register will change the frequency of the I^2C clock
	Concern	Changing the PRER value while the controller is enabled (i.e., CTR.EN = 1) will cause transmission errors
	Mitigation	None. Recommendation: Integration protections should be provided to prevent the PRER register from changing values once the controller is enabled
2	Asset	Bit Command Controller
	Access point	CTR.bit7 register (WISHBONE interface)
	Threat	In the middle of a data transfer on the I^2C, clear the enable bit ("EN") in the controller register (CTR)
	Concern	Toggling the "EN" bit can cause stalls and/or data loss on the I^2C bus
	Mitigation	None. Recommendation: Integration protections should be provided to prevent toggling this bit when data is being transferred

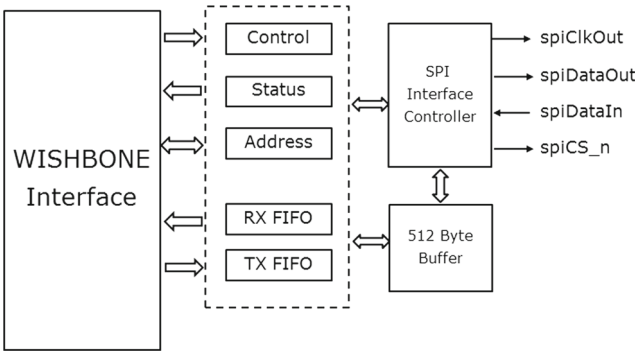


Fig. 6 SPI master controller

Bit Command Controller component, along with the Clock Generator. Since these components are critical for proper operation, they are the assets of the IP. The WISHBONE interface is used to access them. Table 8 is a threat model for the I^2C controller. Again, the identified threats fall within

Table 9 Threat model for SPI master controller

#	Attribute	Definition
1	Asset	SPI interface controller
	Access point	Control register (SPI_CLK_DEL_REG)
	Threat	During step#5 of the write operation, modify the SPI clock delay
	Concern	Modifying the SPI clock frequency could cause data corruption on some SD cards
	Mitigation	None. Recommendation: The integration layer should prevent the SPI clock delay value from changing during a data transfer
2	Asset	SPI Interface Controller
	Access point	TX FIFO (TX_FIFO_CONTROL_REG)
	Threat	During step#5 of the write operation, force the TX FIFO to be clear by setting TX_FIFO_CONTROL_REG=1
	Concern	Clearing the TX FIFO could cause data corruption on some SD cards
	Mitigation	None. Recommendation: The integration layer should disable the TX_FIFO_CONTROL_REG register during a data transfer
3	Asset	SPI Interface Controller
	Access point	TX FIFO (TX_FIFO_DATA_REG)
	Threat	Before step#4 of the write operation, overflow the 512 byte buffer
	Concern	Overflowing the FIFO buffer will cause the buffer to hold invalid data
	Mitigation	None. Recommendation: The integration layer should validate the size of the data before inserting it into the controller

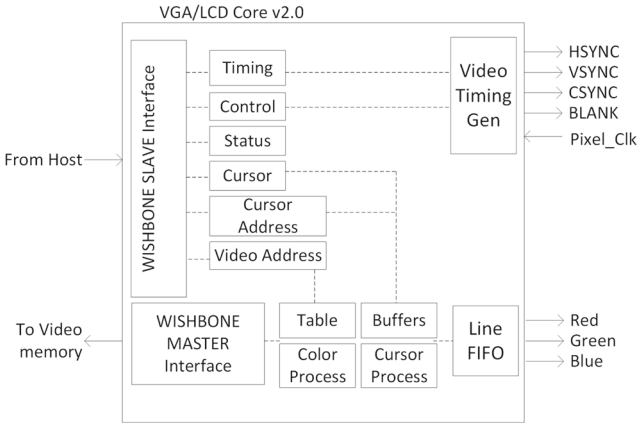


Fig. 7 VGA/LCD controller[4]

the expected behavior of the IP. However, there are two behaviors that an integrating product may want to prevent. Both can be easily mitigated at the integration layer.

4.2 SPI Master Controller

Figure 6 shows the block diagram of the SPI Master Controller IP [15]. A WISHBONE interface exposes configuration, status, and data registers. Since the SPI bus is serial, there is a 512 byte buffer used by the SPI Interface Controller component to serialize data.

There are three basic modes of operation: (1) SD card initialization, (2) SD read, and (3) SD write. The first two modes are fairly benign and not of interest. The write operation, however, has behavior that could be of concern. The basic operational flow for this mode is as follows:

1. Write 512 bytes into TX_DATA_REG
2. Set the SD block address
3. Set the mode to SPI_WRITE_BLOCK for a write operation
4. Start the transaction: SPI_START
5. Wait for transaction to complete: STATUS != BUSY
6. Check for any errors: STATUS = WRITE_NO_ERROR

4.2.1 SPI Master Controller—Threat Model

The functional purpose of this controller is to provide a SPI interface on the WISHBONE bus. For writes, it takes byte data from the internal buffer and serializes it onto the SPI data bus. The SPI Interface Controller component performs this task and is therefore the asset of the IP. The system clock used to drive the IP logic is also essential for proper

[4] http://opencores.org/websvn,filedetails?repname=vga_lcd&path=%2Fvga_lcd%2Ftags%2Frel_19%2Fdoc%2Fvga_core.pdf

operation. However, since this is external to the IP, it is a dependency rather than an asset. Access to the SPI Interface Controller component is provided by the WISHBONE interface. Table 9 is a threat model for the IP. The identified threats are mainly caused by performing incorrect operations during a data transfer. An integrator should consider mitigating these operations since they can be harmful in a product.

4.3 VGA/LCD Controller

Figure 7 shows the block diagram of the VGA/LCD Controller IP [16]. There are two WISHBONE interfaces: (1) a slave interface for the Host to configure and read status of the IP and (2) a master interface that pulls video data from external memory. On the opposite side, there are two line feeds to control either a CRT or LCD monitor. After the timings and modes are configured, the Host will set the starting address where the video data resides in external memory and enable the controller. Once enabled, the IP will start reading video data and outputting it to the appropriate channel.

4.3.1 VGA/LCD Controller—Threat Model

The functional purpose of this IP is to provide a VGA/LCD controller on a WISHBONE interface. Once the controller is configured, the Timing Generator or Line FIFO component, (depending on which is enabled) and the Memory Puller component, which is not shown in Fig. 7, are responsible for outputting the video data to the correct display. Without these components, the controller would not be able to function correctly. Therefore, these three blocks are the assets of the IP. To access these assets, the WISHBONE interfaces are used. Additionally, the pixel clock input into the Timing Generator is considered part of the attack surface. Table 10 lists the threat model for the IP. The only true concern is that this controller can be used to scrape memory. If not restricted, the controller can be configured to read any memory address on the master WISHBONE bus. Therefore, if the product has any sensitive data or secrets in memory, this IP could be used to extract them.

Table 10 Threat model for VGA/LCD controller

#	Attribute	Definition
1	Asset	Memory Puller
	Access point	Slave WISHBONE (video and cursor base address registers (BAR))
	Threat	Setting the BARs allows the IP to read any external memory address
	Concern	An attacker can attach a VGA/LCD-memory converter on the output signals and use the IP to scrape memory by setting the BARs to any memory address
	Mitigation	None. Recommendations: • Put access controls on the BARs restricting it to only trusted components • Provide memory isolation controls on the IP restricting it from sensitive memory ranges

5 Conclusion

The process and methodology defined in this paper adds to the security assurance story for multi-sourced systems by providing a means for trusted IP vendors to provide an industry defined set of security assurance collateral to semiconductor providers who are integrating their cores. Both the SRA and IP threat modeling exercise can be used to identify and address integration security concerns that may be overlooked by other techniques such as formal verification. By examining the functional assets of an IP, unwanted behaviors can be identified and properly mitigated, either within the IP itself or when integrated into a product. As shown in the examples, the unwanted behaviors are often not a result of faulty architecture or design but rather improper use cases. Once identified and properly documented, Integrators can take the necessary steps to prevent these behaviors from occurring in their product, thus improving the overall quality. Additionally, the process can be used by Integrators to assess the competence of an IP vendor for establishing and/or increasing trust. Overall, with minimal investment, this process and methodology can yield higher quality products which benefit both IP provider and Integrator.

Acknowledgments The authors would like to thank our colleagues for their efforts in helping with this paper: Heather Monigan and Kevin Yee.

Appendix A: Security Risk Assessment Questions

This section details the contents of the SRA mentioned in Section 3.1. Each figure consists of at least one top-level question and a path flow that is answer dependent. Additionally, each question has a security concern associated with it and may contain an action and/or recommendation to take.

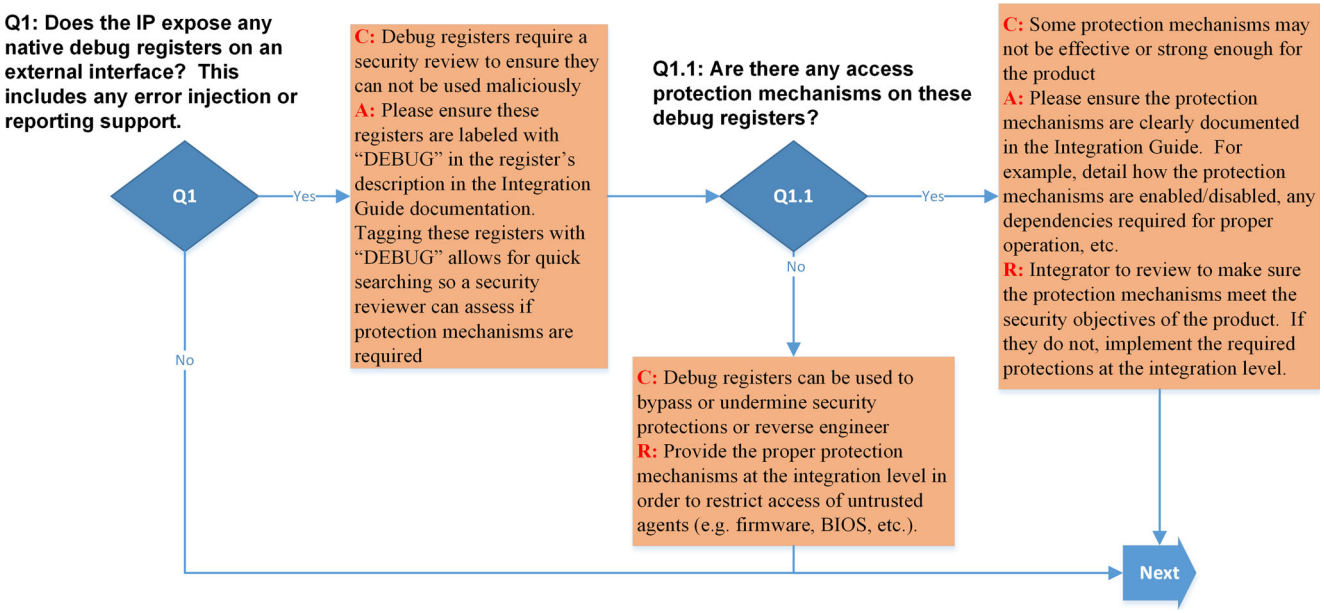


Fig. 8 Debug registers

In the figures, questions are marked with a *Q*. The security concern associated with a question is marked with a *C*. The action that needs to be taken by the IP vendor is marked with an *A*. The recommendation to address the concern is provided to the Integrator and is marked with an *R*. The top-level questions are independent and can be answered in any order. The answers and comments should be documented in a similar format as in Tables 3 and 6.

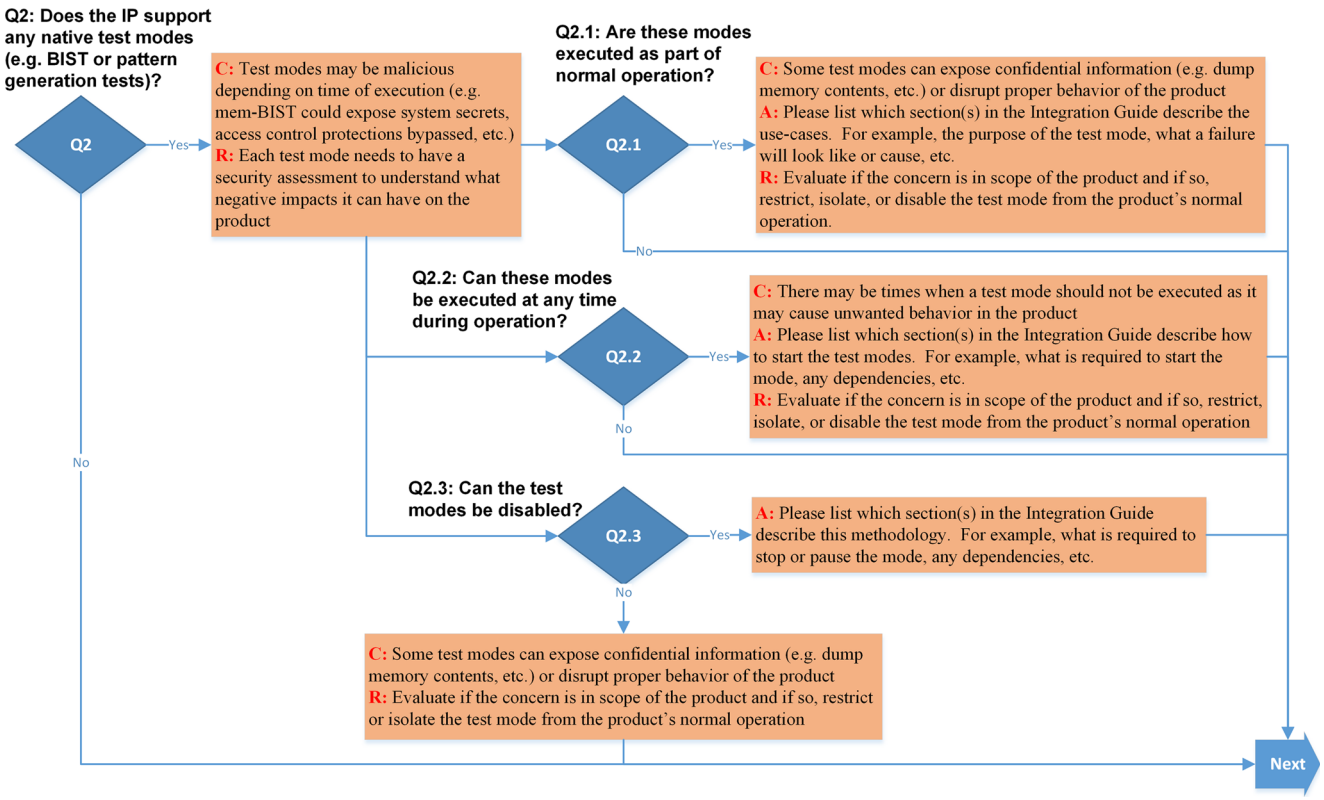


Fig. 9 Test modes

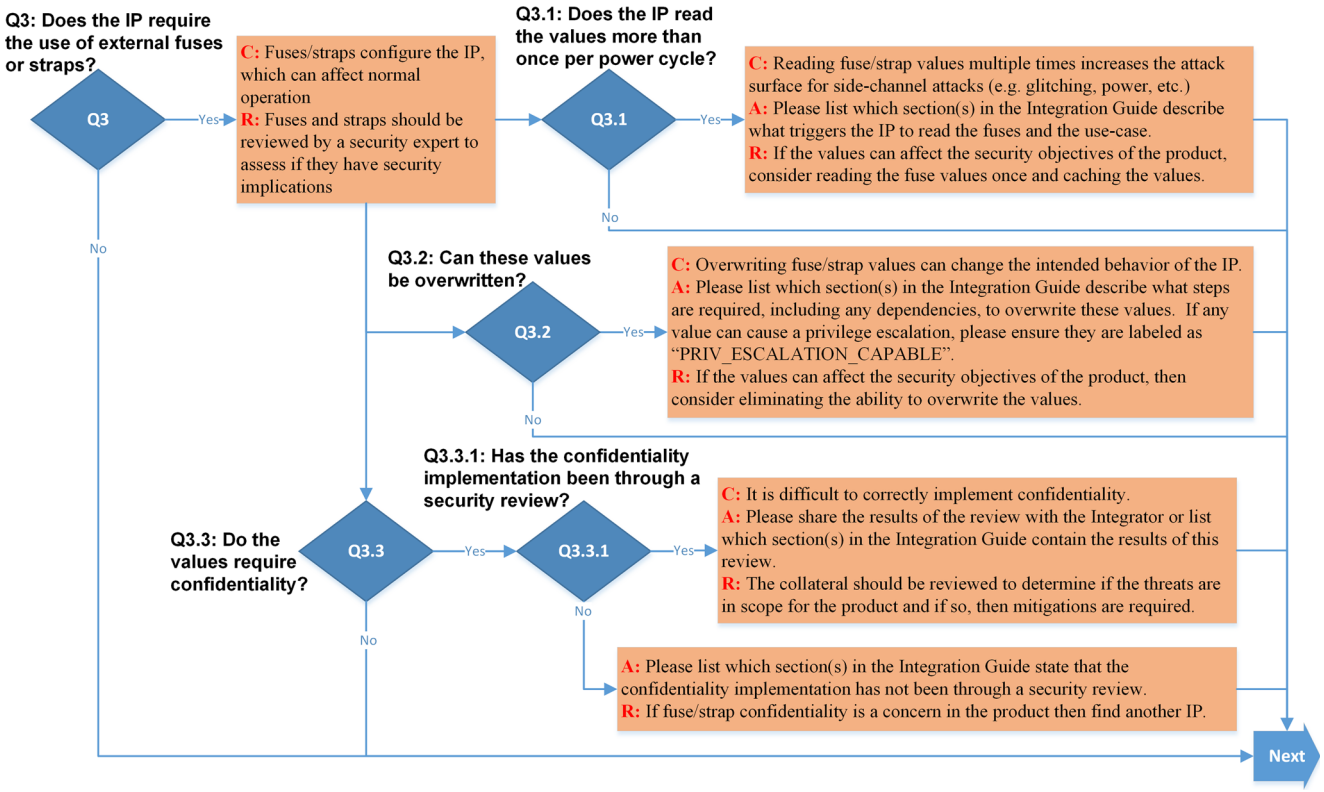


Fig. 10 Fuse and straps

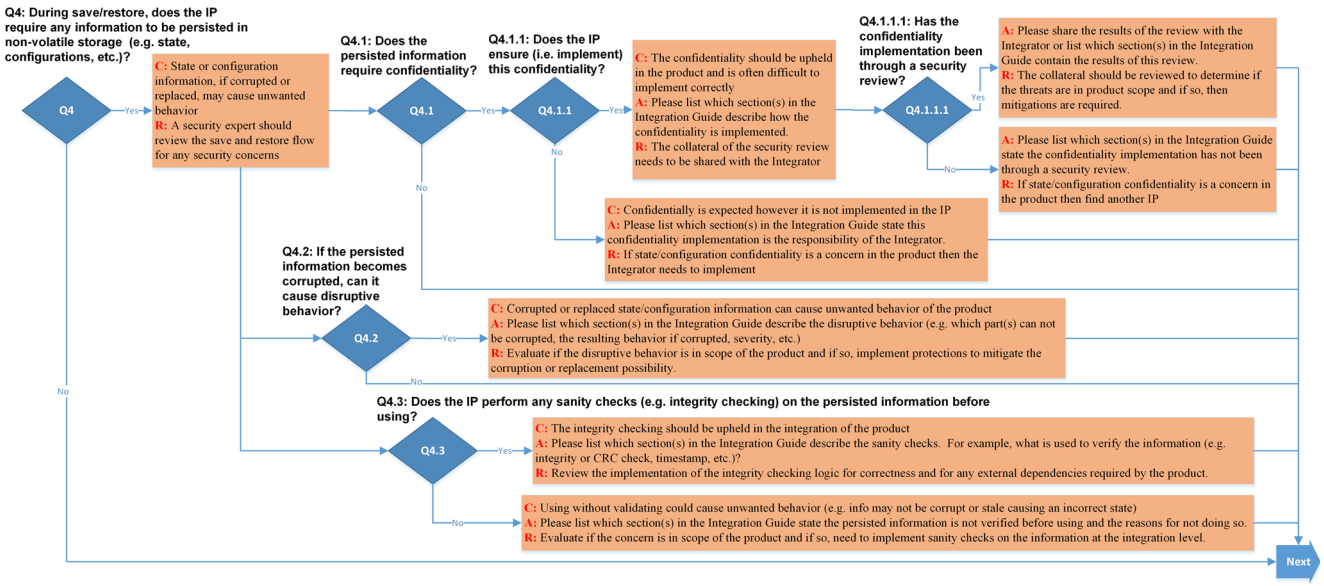


Fig. 11 Save and restore

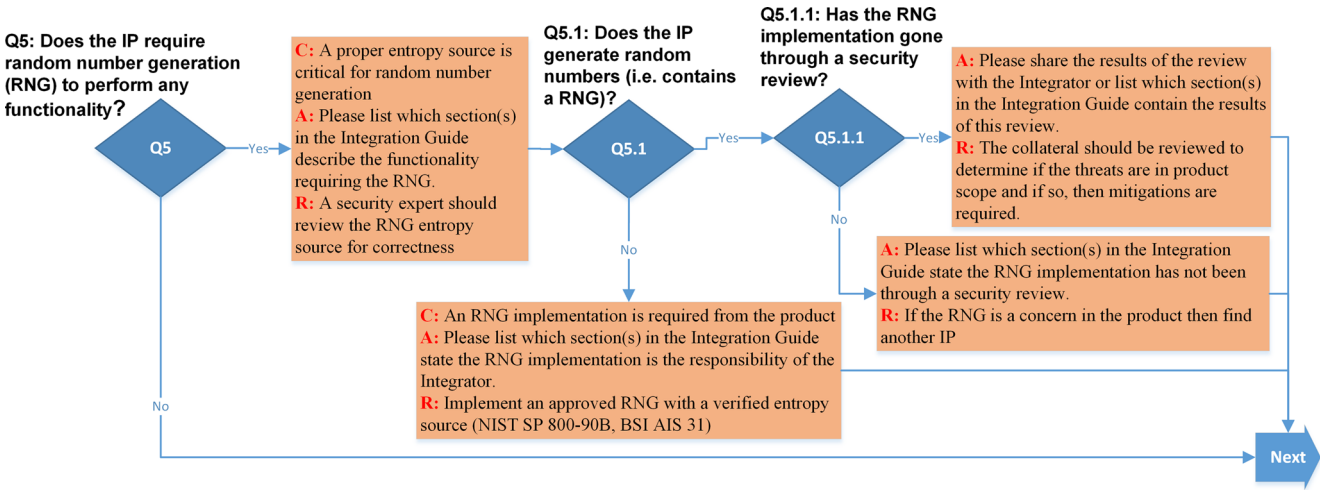


Fig. 12 Random numbers

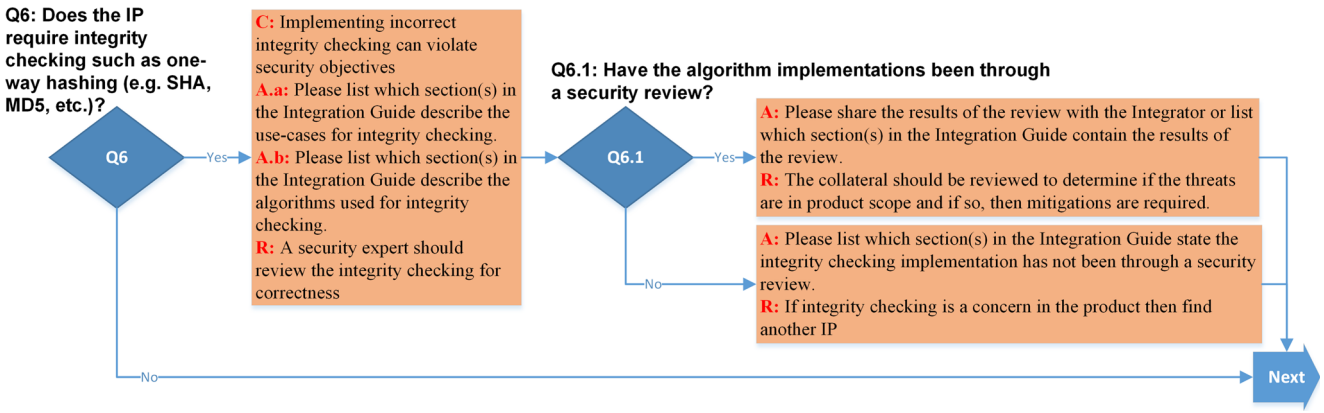


Fig. 13 Integrity checking

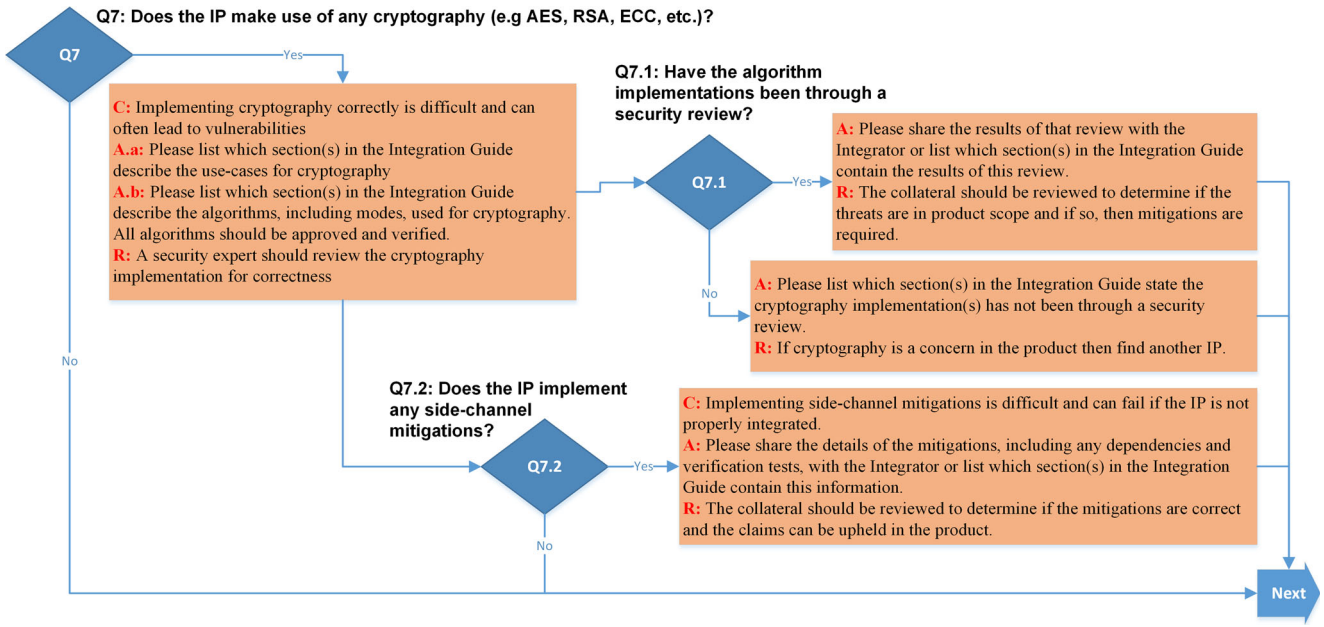


Fig. 14 Cryptography

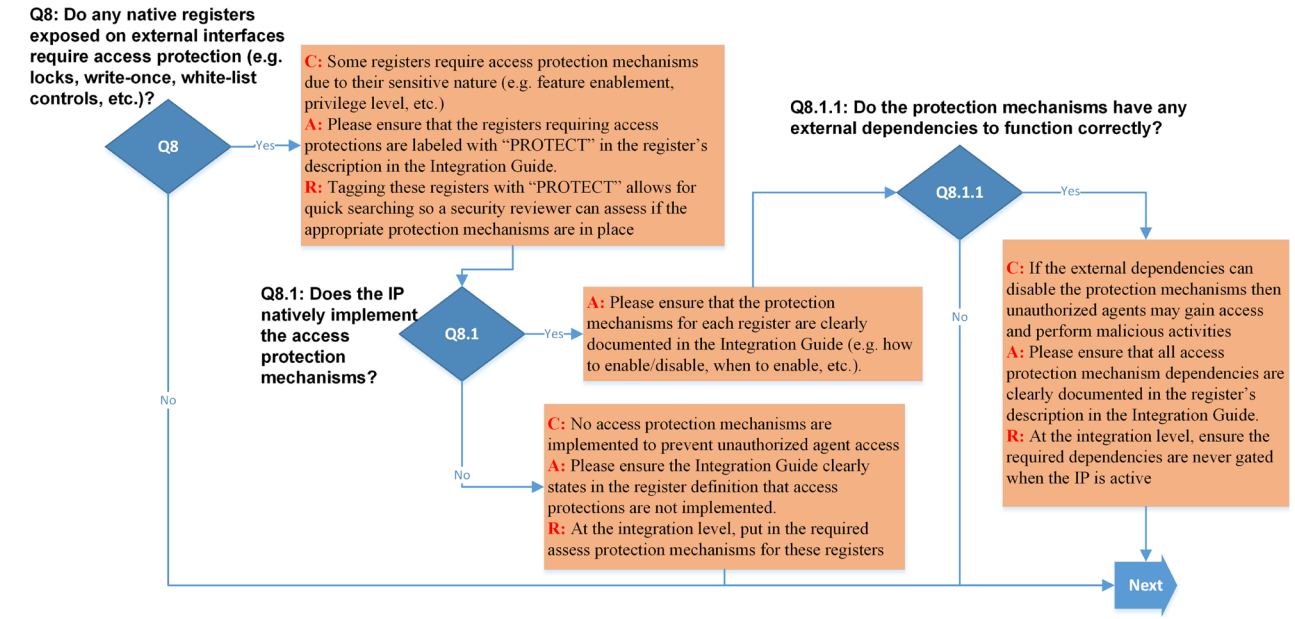


Fig. 15 Access protections

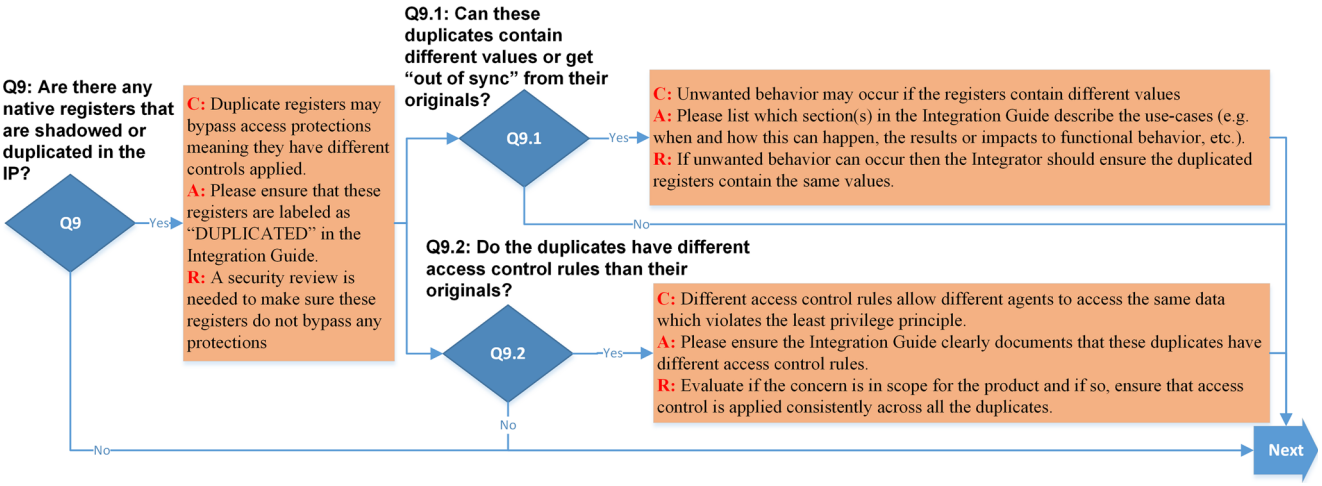


Fig. 16 Duplicate registers

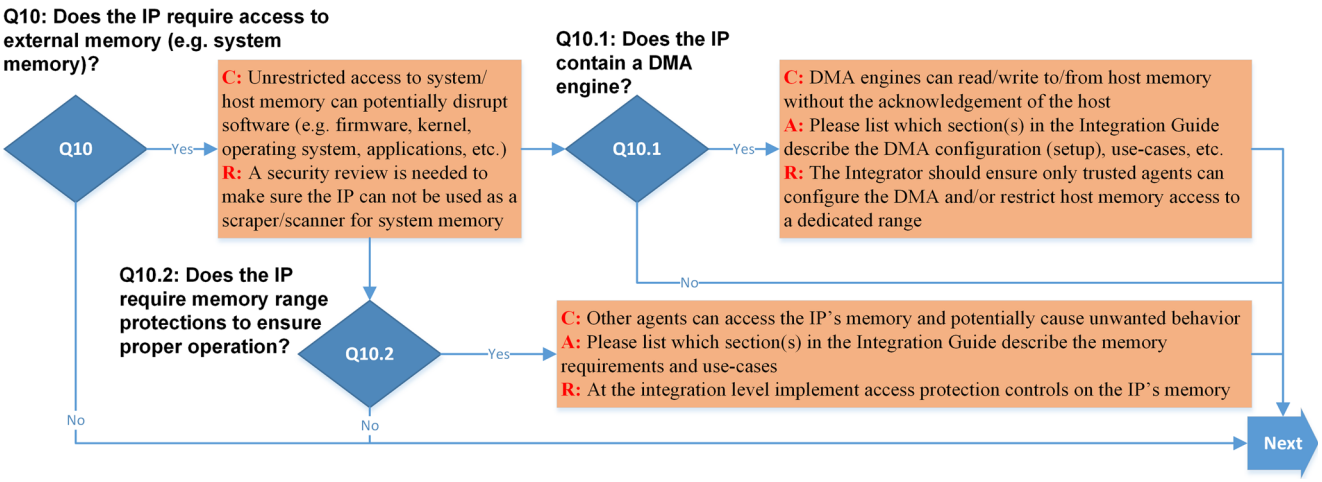


Fig. 17 External memory

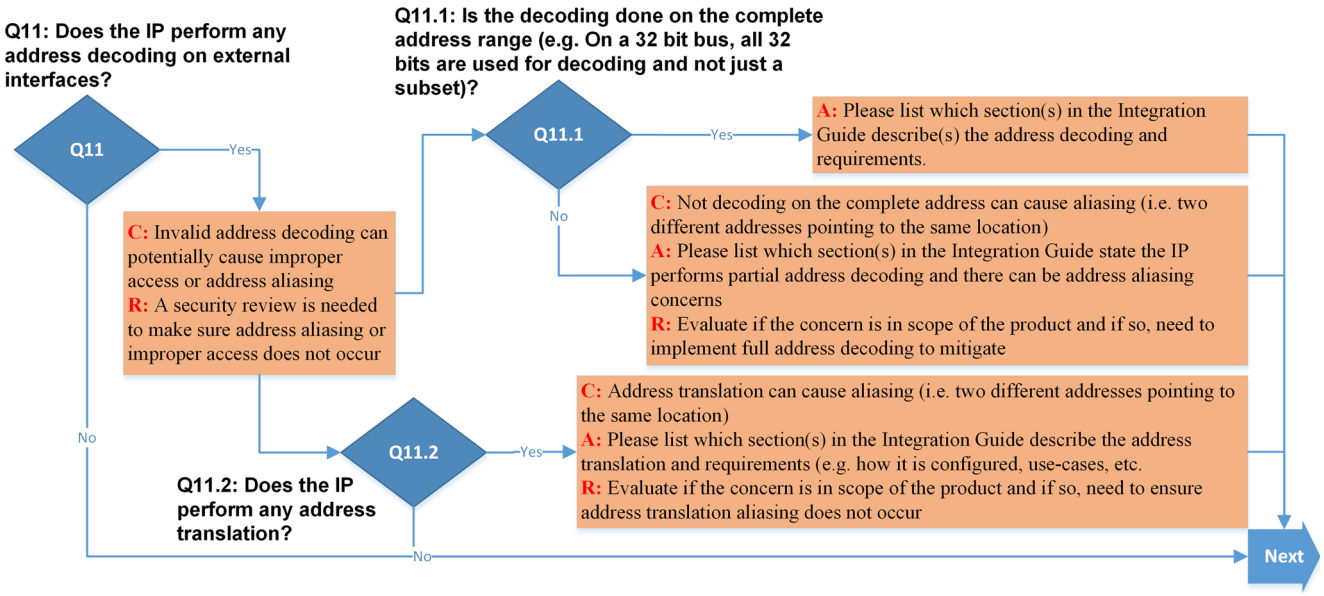


Fig. 18 Address decoding

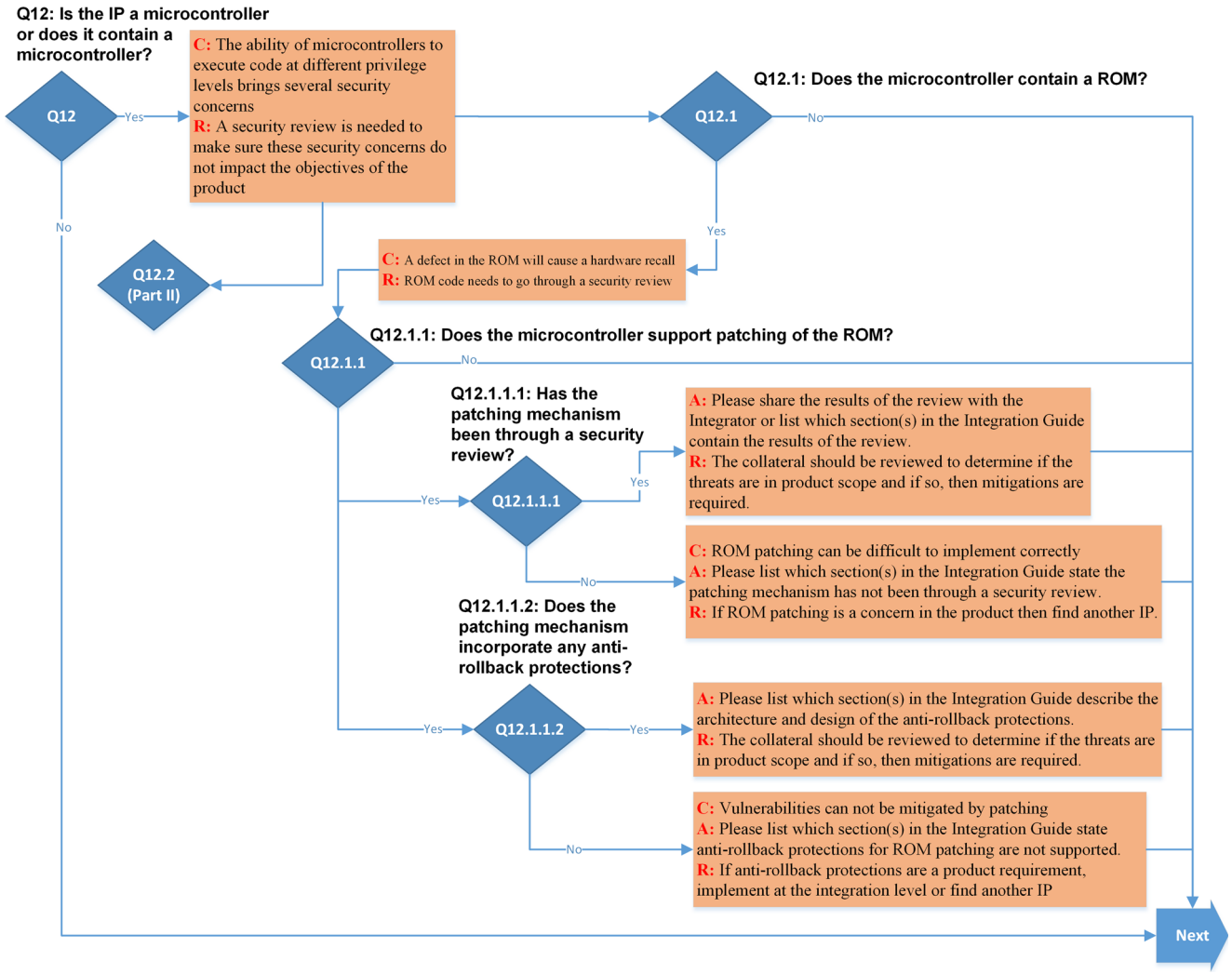


Fig. 19 Microcontroller (ROM)

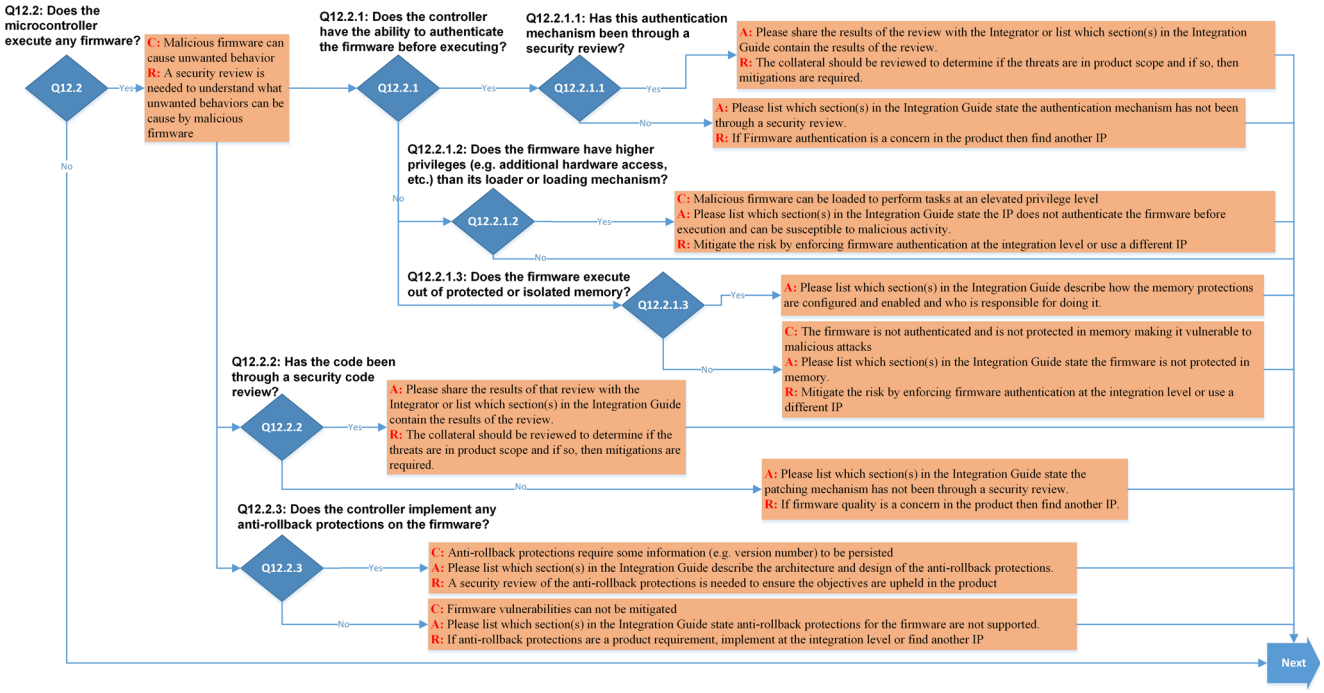


Fig. 20 Microcontroller (Firmware)

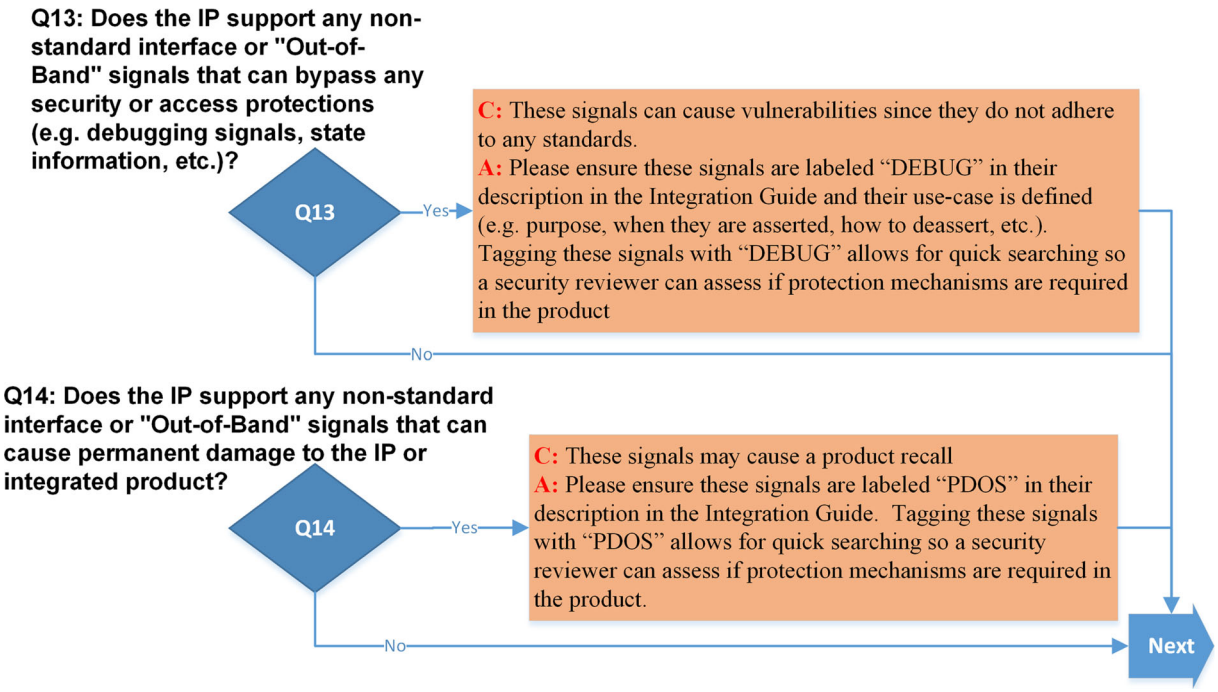


Fig. 21 Non-standard signals

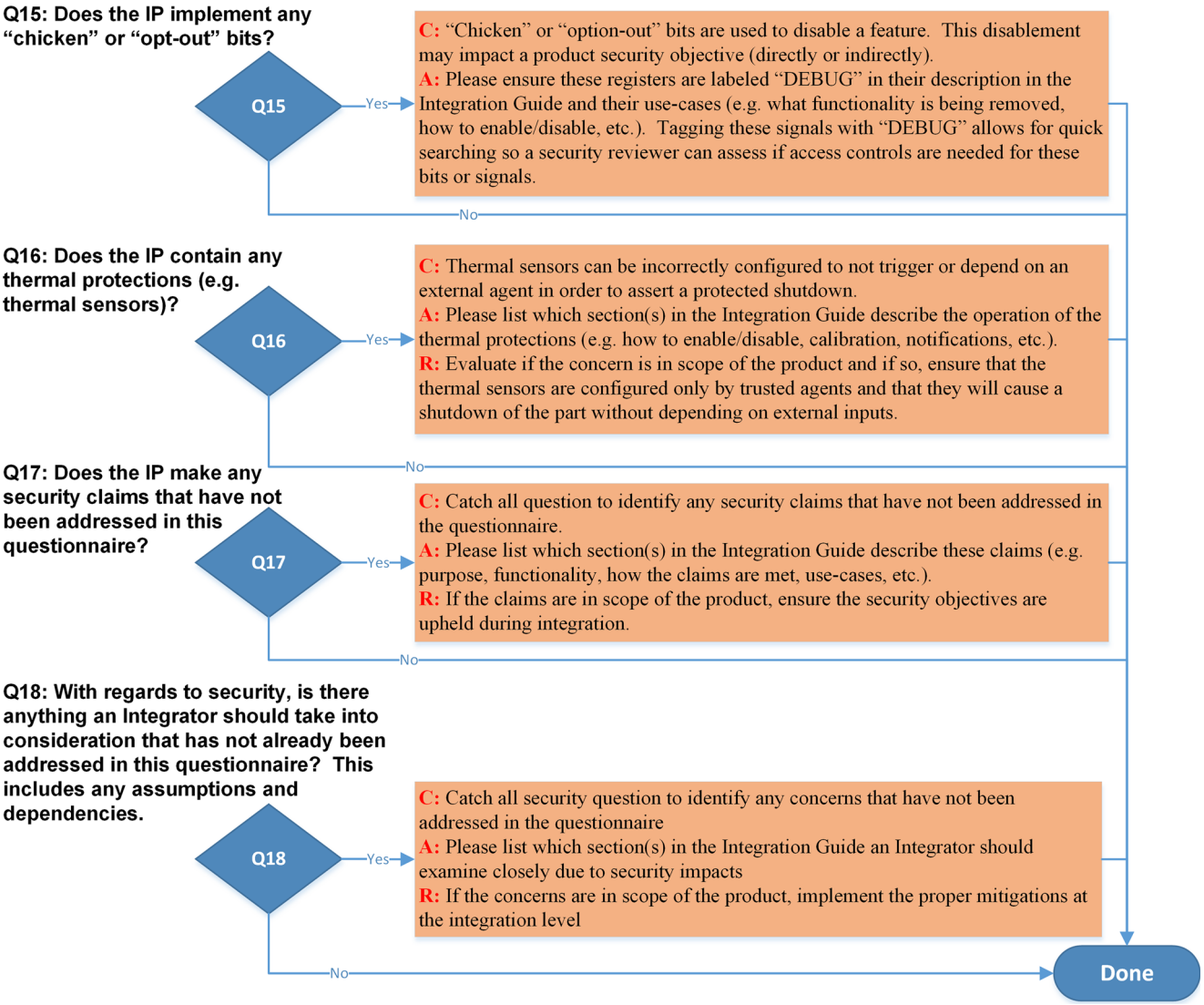


Fig. 22 Miscellaneous

References

1. Lenovo Security Advisory: LEN-2015-010, Retrieved from https://support.lenovo.com/us/en/product_security/superfish
2. BIOS vulnerability targets gigabyte motherboards, hot for security, Retrieved from https://www.hotforsecurity.com/blog/bios-vulnerability-targets-gigabyte-motherboards-14689.html
3. Dell knowledge base, eDellroot certificate, Retrieved from http://www.dell.com/support/article/us/en/19/SLN300321
4. Kemp D, Czub C, Davidov M (2016) Out-of-box exploitation: a security analysis of OEM updaters, duo security. https://duo.com/assets/pdf/out-of-box-exploitation_oem-updaters.pdf
5. Meland PH et al. (2013) The use and usefulness of threats in goal-oriented modelling. availability, reliability and security (ARES). In: 2013 eighth international conference, Sept. 201, Regensburg, Germany, pp 428–436
6. Microsoft. 2010. Security development lifecycle: simplified implementation of the microsoft SDL, http://www.microsoft.com/en-us/download/details.aspx?id=12379
7. Bruza PD, Van der Weide ThP (1993) The semantics of data flow diagrams. In: Proceedings of the international conference on management of data, McGraw-Hill Publishing Company, pp 66–78
8. Sherman B, Wheeler D (2016) Hardware security risk assessment: a case study. In: HOST, 2016, 2016 IEEE international symposium on hardware oriented security and trust (HOST), pp 179–184. doi:10.1109/HST.2016.7495579
9. Texas Tech University, threat_and_risk_modeling, Retrieved from http://discl.cs.ttu.edu/cybersecurity/doku.php?id=threat_and_risk_modeling
10. The Open Web Application Security Project (OWASP), application threat modeling, Retrieved from https://www.owasp.org/index.php/Application_Threat_Modeling
11. Alfredo O RC4 pseudo-random stream generator, OpenCores.org, Retrieved from http://opencores.org/project,rc4-prbs
12. RC4, Wikipedia, The Free Encyclopedia, Retrieved from https://en.wikipedia.org/wiki/RC4
13. Hayes R Computer operating properly, OpenCores.org, http://opencores.org/project,cop

14. Herveille R I2C controller core, OpenCores.org, http://opencores.org/project,i2c
15. Fielding S SD/MMC Controller, OpenCores.org, http://opencores.org/project,spimaster
16. Herveille R VGA/LCD Controller, OpenCores.org, http://opencores.org/project,vga_lcd
17. Shostack A (2014) Threat modeling: designing for security. Wiley, Indianapolis, Indiana
18. Xiao K, Nahiyan A, Tehranipoor M (2016) Security rule checking in IC design. Computer 49(8):54–61. doi:10.1109/MC.2016.226
19. Nahiyan A, Xiao K, Yang K, Jin Y, Forte D, Tehranipoor M (2016) AVFSM: a framework for identifying and mitigating vulnerabilities in FSMs. In: 2016 53nd ACM/EDAC/IEEE design automation conference (DAC), Austin, TX, pp 1–6. doi:10.1145/2897937.2897992